**FASE 2**

**DISEÑO E IMPLEMENTACIÓN DEL PROYECTO**

**Capstone**

**MEDINFO**

Escuela de Informática y Telecomunicaciones

Noviembre 2025

1. Identificación del Proyecto

| **Nombre del Proyecto** |
| --- |
| MedInfo |

1. Integrantes del Equipo de Trabajo

| **N°** | **Rut** | **Apellidos** | **Nombres** |
| --- | --- | --- | --- |
| 1 | 20.497.699-6 | Guzmán Aguirre | Mauricio Ignacio |
| 2 | 17.148.513-4 | Rodríguez Flores | Matías Sebastián |
| 3 | 19.974.960-9 | Díaz | Rafael |

1. Registro de Control de Cambios

| **Revisión** | **Fecha** | **Páginas** | **Descripción del Cambio** | **Autor** |
| --- | --- | --- | --- | --- |
| 1 | 09-09-25 | Todas | Cambios según primera revisión docente (alcances y objetivos). | DDI |
| 2 | 22-09-25 | Todas | Avance | DDI |
| 3 | 06-10-25 | Todas | Avance | DDI |
| 4 | 30-10-25 | Todas | Final | DDI |
| 5 | 18-11-25 | Todas | Final | DDI |

# 

# ÍNDICE DE CONTENIDO

[ÍNDICE DE CONTENIDO 3](#_heading=h.6cz9alva0h1i)

[LISTA DE TABLAS 4](#_heading=h.guidrknf6z5k)

[LISTA DE FIGURAS 5](#_heading=h.nkftntycfwl1)

[LISTA DE DIAGRAMAS 6](#_heading=h.rrknp2yglxyo)

[GLOSARIO 7](#_heading=h.4rxyuxaf3rc6)

[1 Diseño e Implementación del Proyecto 10](#_heading=h.irxv57v79q7q)

[1.1 Resumen 10](#_heading=h.z6m7336808hn)

[1.2 Abstract 12](#_heading=)

[1.3 Introducción 14](#_heading=h.ilquxk8ouo6k)

[1.4 Problemática a solucionar o necesidad a satisfacer 16](#_heading=h.8ubymcwrnmfn)

[1.5 Objetivos del Proyecto (general y específicos) 19](#_heading=h.l2egz5hb0go)

[1.6 Competencias del Perfil de Egreso 21](#_heading=h.k1baq7di2b9d)

[1.7 Asignación de roles 22](#_heading=h.xwcs9tsjlz7g)

[1.8 Metodología utilizada en el Proyecto 23](#_heading=h.kwiytarj59vz)

[1.9 Creación de cronograma asociado al Proyecto (Carta Gantt) 27](#_heading=h.eha58oq18wf3)

[1.10 Riesgos Asociados al Proyecto 27](#_heading=h.jh48jj1rv41y)

[1.11 Implementación del Proyecto 30](#_heading=h.esinc8ydkd7q)

[Diseño y arquitectura de la solución 30](#_heading=h.vauqg18xjxc3)

[Requerimientos técnicos 34](#_heading=h.rou6phmcgmjo)

[Desarrollo de la solución 36](#_heading=h.y6f0invl1nlx)

[Resultados de la solución 40](#_heading=h.azr90peeo6vz)

[1.12 Definición de Recursos y Costos asociados al Proyecto 45](#_heading=h.9ba5vfipqp6e)

[CONCLUSIÓN 47](#_heading=h.eyqwopna13u2)

[BIBLIOGRAFÍA 49](#_heading=h.m6noryc08b7b)

[ANEXOS 50](#_heading=)

# LISTA DE TABLAS

| No. | Nombre de la Tabla | Ubicación |
| --- | --- | --- |
| Tabla 1 | Matriz de Riesgos y Planes de Contingencia | [Pág. 25,26](#_heading=h.dvcmny2mlx2r) |
| Tabla 2 | Requerimientos Técnicos (Stack Tecnológico de la Fase 2) | [Pág. 31](#_heading=h.rou6phmcgmjo) |
| Tabla 3 | Costo de Recursos Humanos (Fase de Desarrollo) | [Pág. 38](#_heading=h.at6c6cncokaj) |
| Tabla 4 | Costos Tecnológicos y Operacionales (Estimación Mensual) | [Pág. 39](#_heading=h.5flnra1hwzg6) |

# 

# 

# 

# 

# LISTA DE FIGURAS

| No. | Nombre de la Figura | Ubicación |
| --- | --- | --- |
| Figura 1 | Prueba de Autenticación Exitosa | [Anexo C](http://anexos/Anexo_C.1.png) |
| Figura 2 | Acceso Autorizado (Uso del JWT) | [Anexo C](http://anexos/Anexo_C.2.png) |
| Figura 3 | Denegación de Acceso (Error 401) | [Anexo C](http://anexos/Anexo_C.3.png) |
| Figura 4 | Creación de Recurso (Registro en DB Neon) | [Anexo C](http://anexos/Anexo_C.4.png) |
|  |  |  |

# 

# 

# 

# 

# 

# LISTA DE DIAGRAMAS

| No. | Nombre del Diagrama | Ubicación |
| --- | --- | --- |
| Diagrama 1 | Diagrama de Arquitectura Lógica (3 Capas) | [Anexo B](http://anexos/Anexo_B_Diagrama_de_Arquitectura.png) |
| Diagrama 2 | Diagrama Entidad-Relación (DER) del Modelo de Datos | [Anexo A](http://anexos/Anexo_A_Diagrama_ER.png) |
|  |  |  |

# GLOSARIO

**API REST** Application Programming Interface (Interfaz de Programación de Aplicaciones) Representational State Transfer. Es un conjunto de reglas y protocolos que permite a diferentes sistemas (como la app móvil y el backend) comunicarse a través del protocolo HTTP.

**Alembic** Herramienta de código abierto utilizada en proyectos Python (como el Backend Flask) para gestionar **migraciones de bases de datos**. Permite al equipo de desarrollo evolucionar el esquema de la base de datos (PostgreSQL) de manera controlada y reversible.

**Angular** Framework de código abierto basado en TypeScript, mantenido por Google. Utilizado para construir la interfaz de usuario del Dashboard Médico (Frontend Web), ideal para aplicaciones empresariales grandes y complejas.

**Microsoft Azure App Service (Linux) Microsoft Azure Elastic Compute Cloud.** Servicio de computación en la nube que proporciona capacidad de servidor redimensionable. Utilizado en MedInfo para el almacenamiento persistente y escalable de archivos clínicos y el despliegue del Backend.

**Backend** La parte del sistema que no es visible para el usuario final. En MedInfo, es el servidor central (Flask) que contiene la lógica de negocio, gestiona los datos y expone las APIs.

**CI/CD** Continuous Integration / Continuous Delivery. Conjunto de prácticas que automatizan las etapas de desarrollo de software, desde la integración de código hasta el despliegue en producción, asegurando actualizaciones rápidas y sin interrupciones.

**Cifrado en Reposo** Proceso de codificación de datos cuando están almacenados (en la base de datos o en el servidor de archivos) para proteger su confidencialidad.

**Cifrado en Tránsito** Proceso de codificación de datos mientras son transferidos a través de la red (garantizado por HTTPS), evitando que sean interceptados.

**CORS** Cross-Origin Resource Sharing. Mecanismo de seguridad que controla qué recursos (Frontend) tienen permiso para hacer peticiones al servidor (Backend) desde un dominio diferente.

**Dashboard** Interfaz gráfica de usuario que presenta la información de manera centralizada y organizada (ej. estadísticas, accesos rápidos). En MedInfo, es la interfaz principal para los médicos.

**Flask** Micro-framework de desarrollo web en Python, liviano y modular. Utilizado para construir el Backend y las APIs REST de MedInfo.

**Frontend:** La parte del sistema con la que el usuario interactúa directamente (la interfaz de usuario). En MedInfo, son la App Móvil (Ionic) y el Dashboard Web (Angular).

**HTTPS** Hypertext Transfer Protocol Secure. Protocolo de comunicación web que garantiza la seguridad y la privacidad de los datos mediante el cifrado. Es esencial para cumplir con ISO 27001.

**Ionic** Framework de código abierto que permite el desarrollo de aplicaciones móviles híbridas (iOS/Android) utilizando tecnologías web como Angular.

**ISO 27001** Norma internacional que especifica los requisitos para establecer, implementar, mantener y mejorar un Sistema de Gestión de la Seguridad de la Información (SGSI), fundamental para la protección de datos clínicos.

**JWT** JSON Web Token. Estándar compacto y seguro para transmitir información entre partes como un objeto JSON. Utilizado para la autenticación y autorización de usuarios en las APIs.

**Lazy Loading** Técnica de optimización de rendimiento en aplicaciones web donde los módulos o recursos se cargan sólo cuando son necesarios, reduciendo el tiempo de carga inicial.

**Metodología Ágil** Enfoque de gestión de proyectos que prioriza la entrega continua de valor, la flexibilidad ante los cambios y la colaboración del equipo (ej. Scrum).

**Modelo Entidad-Relación (E-R)** Herramienta de modelado conceptual de datos que ilustra las entidades principales (tablas) de la base de datos y cómo se relacionan entre sí. Es el plano fundamental de la estructura de datos que garantiza la integridad y consistencia de la información clínica.

**Monorepo** Estrategia de gestión de código fuente en la que el código para múltiples proyectos (como el Dashboard Angular, la App Ionic y el Backend Flask) se almacena en un único repositorio unificado. Facilita la reutilización de código y la gestión de dependencias.

**Azure Database for PostgreSQL** Plataforma de base de datos serverless que aloja PostgreSQL y ofrece escalabilidad automática, garantizando la disponibilidad de los datos.

**PostgreSQL** Sistema de gestión de base de datos relacional de código abierto. Elegido por MedInfo por su robustez, integridad de datos y características avanzadas de seguridad.

**Scrum** Framework ágil que utiliza ciclos de trabajo cortos llamados Sprints para gestionar y entregar productos complejos. Es la metodología elegida para el desarrollo de MedInfo.

**WebSockets** Protocolo de comunicación que proporciona un canal de comunicación bidireccional y persistente sobre una única conexión TCP, usado para el chat en tiempo real y las notificaciones instantáneas.

# **Diseño e Implementación del Proyecto**

## **Resumen**

El sistema MedInfo constituyó una plataforma médica integral que facilitó la gestión clínica y la comunicación entre profesionales de la salud y sus pacientes a través de dos aplicaciones independientes pero interconectadas. La arquitectura del sistema separó completamente las interfaces de usuarios según su rol, implementando un dashboard profesional para médicos desarrollado en Angular y una aplicación móvil para pacientes construida con Ionic; ambas interfaces compartieron un backend común desarrollado en Flask que expuso APIs REST y se conectó a una base de datos PostgreSQL alojada en Azure Database for PostgreSQL.

La aplicación para médicos (**medinfo-doctor-dashboard**) presentó una interfaz de escritorio optimizada para la gestión clínica con un dashboard que centralizó estadísticas relevantes, acceso a historiales de pacientes, gestión de citas y exámenes, archivos médicos categorizados y un sistema de mensajería en tiempo real. La interfaz siguió principios de diseño centrado en el usuario con navegación intuitiva, minimización de clics y feedback visual inmediato; implementó *lazy loading*, paginación eficiente y uso estratégico de caché para optimizar el rendimiento incluso con grandes volúmenes de datos médicos.

El módulo de gestión de archivos médicos permitió subir, categorizar y compartir documentación clínica crítica como informes médicos, recetas, resultados de exámenes, imágenes diagnósticas y documentos generales; el sistema utilizó almacenamiento híbrido que combinó almacenamiento local temporal cuando no había conexión y sincronización con Microsoft Azure App Service (Linux) para persistencia en la nube, sin límites de tamaño para adaptarse a archivos médicos de alta resolución. Las búsquedas avanzadas por paciente, tipo de documento, fecha y contenido facilitaron la recuperación rápida de información clínica relevante.

El sistema de agendamiento y horarios incorporó un calendario interactivo donde los médicos cargaron su disponibilidad mediante archivos CSV/Excel y también realizaron ajustes directamente en la interfaz gráfica; los pacientes visualizaron los horarios disponibles y solicitaron citas que requirieron aprobación médica, con notificaciones automáticas para ambas partes sobre confirmaciones, cancelaciones o reprogramaciones. La visualización de la agenda integró códigos de colores para distinguir tipos de consulta, estados y prioridades.

La comunicación médico–paciente se implementó mediante un sistema de chat en tiempo real con salas individuales para cada relación médico–paciente; los mensajes incluyeron indicadores de lectura, priorización de urgencias y capacidad para adjuntar archivos relevantes como fotografías de síntomas o documentación complementaria. El chat implementó *WebSockets* para comunicación instantánea y notificaciones *push* en ambas aplicaciones.

El módulo de exámenes médicos permitió a los profesionales ordenar estudios, hacer seguimiento por estados (ordenado, programado, completado) y almacenar resultados con interpretaciones asociadas; los pacientes recibieron notificaciones sobre nuevos exámenes ordenados, preparaciones necesarias y disponibilidad de resultados, pudiendo acceder a ellos directamente desde la aplicación móvil.

La seguridad del sistema cumplió rigurosamente con la norma ISO 27001, implementando autenticación robusta con JWT (*JSON Web Tokens*), roles estrictamente separados, control de acceso granular por recursos, cifrado en tránsito mediante HTTPS y cifrado en reposo para datos sensibles. Los eventos críticos como intentos de acceso, consulta de archivos confidenciales y modificaciones de datos clínicos se registraron en un sistema de auditoría completo.

El backend implementó validación exhaustiva de entradas, manejo estructurado de errores, *rate limiting* para prevenir ataques y políticas CORS restrictivas; todos los *endpoints* estuvieron protegidos mediante *middlewares* de autenticación que validaron tokens y permisos antes de autorizar operaciones. La base de datos PostgreSQL se mantuvo actualizada con esquemas seguros, índices optimizados y respaldos automatizados.

Para asegurar la disponibilidad del servicio incluso en condiciones de conectividad limitada, ambas aplicaciones implementaron estrategias de caché y almacenamiento local temporal que permitieron consultar información previamente accedida y crear nuevos registros que se sincronizan automáticamente cuando se restableció la conexión. Este enfoque resultó particularmente valioso en entornos clínicos con infraestructura de red variable.

El sistema **MedInfo** fue diseñado con capacidad de escalabilidad horizontal, permitiendo aumentar su capacidad mediante la adición de instancias adicionales del backend detrás de un balanceador de carga; el código siguió patrones de diseño como inyección de dependencias, separación de responsabilidades y contratos de interfaz claros que facilitaron su mantenimiento y evolución futura. Las actualizaciones se realizaron con estrategias de despliegue continuo que minimizaron el tiempo de inactividad.

En conclusión, **MedInfo** representó una solución tecnológica completa que abordó las necesidades específicas del sector salud con interfaces diferenciadas según el rol, seguridad alineada con estándares internacionales, procesamiento eficiente de datos clínicos y comunicación fluida entre médicos y pacientes; su arquitectura modular y enfoque en la experiencia de usuario permitieron una adaptación natural a los flujos de trabajo clínicos existentes, mejorando la productividad profesional y la satisfacción del paciente.

## Abstract

The MedInfo system constituted an integrated medical platform that facilitated clinical management and communication between healthcare professionals and their patients through two independent yet interconnected applications. The system’s architecture completely separated user interfaces according to their roles, implementing a professional dashboard for doctors developed in Angular and a mobile application for patients built with Ionic. Both interfaces shared a common backend developed in Flask, which exposed REST APIs and connected to a PostgreSQL database hosted on Azure Database for PostgreSQL.

The application for doctors (**medinfo-doctor-dashboard**) presented a desktop interface optimized for clinical management, with a dashboard that centralized relevant statistics, access to patient histories, appointment and examination management, categorized medical records, and a real-time messaging system. The interface followed user-centered design principles with intuitive navigation, minimal clicks, and immediate visual feedback. It implemented lazy loading, efficient pagination, and strategic cache use to optimize performance even with large volumes of medical data.

The medical records management module allowed uploading, categorizing, and sharing critical clinical documentation such as medical reports, prescriptions, exam results, diagnostic images, and general documents. The system used a hybrid storage model combining temporary local storage when offline and synchronization with Microsoft Azure App Service (Linux) for cloud persistence, with no size limits to accommodate high-resolution medical files. Advanced searches by patient, document type, date, and content facilitated quick retrieval of relevant clinical information.

The scheduling and appointments system incorporated an interactive calendar where doctors uploaded their availability via CSV/Excel files and also made adjustments directly within the graphical interface. Patients viewed available time slots and requested appointments that required medical approval, with automatic notifications for both parties regarding confirmations, cancellations, or rescheduling. The calendar displays integrated color codes to distinguish consultation types, statuses, and priorities.

Doctor–patient communication was implemented through a real-time chat system with individual rooms for each doctor–patient relationship. Messages included read indicators, urgency prioritization, and the ability to attach relevant files such as photos of symptoms or complementary documentation. The chat used WebSockets for instant communication and push notifications in both applications.

The medical exams module allowed professionals to order studies, track them by status (ordered, scheduled, completed), and store results with associated interpretations. Patients received notifications about newly ordered exams, required preparations, and result availability, allowing them to access this information directly from the mobile app.

The system’s security rigorously complied with the ISO 27001 standard, implementing robust authentication with JWT (JSON Web Tokens), strictly separated roles, granular access control by resource, encrypted transmission via HTTPS, and data-at-rest encryption for sensitive information. Critical events such as access attempts, confidential file consultations, and clinical data modifications were logged in a comprehensive audit system.

The backend implemented thorough input validation, structured error handling, rate limiting to prevent attacks, and restrictive CORS policies. All endpoints were protected through authentication middlewares that validated tokens and permissions before authorizing operations. The PostgreSQL database remained updated with secure schemas, optimized indexes, and automated backups.

To ensure service availability even under limited connectivity conditions, both applications implemented caching and temporary local storage strategies, which allowed users to view previously accessed information and create new records that synchronized automatically once the connection was restored. This approach proved particularly valuable in clinical environments with variable network infrastructure.

The **MedInfo** system was designed with horizontal scalability capabilities, allowing capacity to be increased by adding additional backend instances behind a load balancer. The code followed design patterns such as dependency injection, separation of concerns, and clear interface contracts, which facilitated maintenance and future evolution. Updates were deployed using continuous delivery strategies that minimized downtime.

In conclusion, **MedInfo** represented a comprehensive technological solution that addressed the specific needs of the healthcare sector with role-based interfaces, security aligned with international standards, efficient clinical data processing, and seamless communication between doctors and patients. Its modular architecture and focus on user experience allowed a natural adaptation to existing clinical workflows, improving professional productivity and patient satisfaction.

## Introducción

El sector de la salud enfrenta desafíos crecientes en la gestión de información clínica y comunicación eficiente entre profesionales médicos y pacientes, especialmente en el contexto post-pandémico donde la digitalización de servicios sanitarios se ha convertido en una necesidad crítica más que en una opción. Las instituciones de salud requieren soluciones tecnológicas que no solo faciliten el acceso y gestión de historiales médicos, sino que también cumplan con rigurosos estándares de seguridad y privacidad de datos, manteniendo la eficiencia operacional y mejorando la experiencia del paciente.

En Chile, el sistema de salud presenta particularidades específicas que demandan herramientas adaptadas a la realidad local; los centros médicos enfrentan volúmenes crecientes de información clínica, necesidad de coordinación entre especialistas, gestión de citas en contextos de alta demanda y comunicación oportuna con pacientes que cada vez más demandan acceso inmediato a sus datos de salud. La brecha digital existente entre diferentes grupos poblacionales requiere interfaces diferenciadas que atiendan tanto a profesionales médicos que necesitan herramientas complejas de gestión como a pacientes que requieren acceso simple e intuitivo a su información personal de salud.

La fragmentación de sistemas existentes en muchas instituciones genera ineficiencias operacionales, duplicación de esfuerzos y riesgos de seguridad por el uso de múltiples plataformas desconectadas; los profesionales médicos dedican tiempo considerable a tareas administrativas que podrían automatizarse, mientras los pacientes experimentan demoras en acceso a resultados, dificultades para agendar citas y comunicación limitada con sus médicos tratantes fuera del horario de consulta.

El proyecto MedInfo surge como respuesta a esta problemática, diseñado específicamente para implementarse en centros médicos, clínicas privadas y consultorios que buscan modernizar sus procesos de gestión clínica mediante una plataforma integral que centralice la información médica, facilite la comunicación profesional-paciente y optimice los flujos de trabajo administrativos. El sistema está concebido para operar en entornos donde la seguridad de la información constituye una prioridad absoluta, cumpliendo con normativas internacionales como ISO 27001 y adaptándose a los requerimientos específicos del marco regulatorio chileno en materia de protección de datos personales y confidencialidad médica.

La implementación del sistema contempla su despliegue en infraestructura de nube mediante Microsoft Azure App Service (Linux), permitiendo escalabilidad según el crecimiento institucional y asegurando disponibilidad del servicio con capacidad de respaldo y recuperación ante desastres; esta arquitectura facilita el acceso remoto necesario para atención médica a distancia y permite la integración gradual con sistemas hospitalarios existentes sin interrumpir operaciones críticas.

El contexto actual de transformación digital en salud, acelerado por la experiencia de telemedicina durante la pandemia, crea un ambiente propicio para la adopción de plataformas como MedInfo que combinan gestión clínica tradicional con comunicación moderna; las instituciones de salud reconocen la necesidad de herramientas que no solo digitalicen procesos existentes sino que rediseñar completamente la experiencia de atención, priorizando la satisfacción del paciente y la eficiencia del profesional médico.

La diferenciación por roles que implementa MedInfo responde directamente a las necesidades específicas identificadas en el sector; los médicos requieren interfaces complejas con acceso a múltiples funcionalidades simultáneas, capacidad de gestión de grandes volúmenes de información y herramientas de análisis, mientras los pacientes necesitan interfaces simples, claras e intuitivas que faciliten el acceso a su información personal sin complejidad técnica innecesaria.

En este contexto, MedInfo se posiciona como una solución integral que aborda las necesidades actuales del sector salud mediante tecnología moderna, seguridad robusta y diseño centrado en el usuario, preparada para implementarse en instituciones que buscan liderar la transformación digital de sus servicios médicos y mejorar significativamente tanto la experiencia del paciente como la productividad del equipo médico.

## Problemática a solucionar o necesidad a satisfacer

**Características de la Problemática**El sector salud chileno enfrenta múltiples desafíos tecnológicos y operacionales que impactan directamente la calidad de atención médica y la eficiencia de las instituciones sanitarias.

La problemática principal radica en la fragmentación de sistemas de información, comunicación deficiente entre médicos y pacientes, y procesos administrativos obsoletos que generan ineficiencias operacionales significativas y afectan la experiencia del usuario final.

La gestión de información clínica presenta serias limitaciones en la mayoría de centros médicos chilenos. Los registros médicos se mantienen frecuentemente en formatos físicos o sistemas digitales desconectados, generando dificultades para acceder a historiales completos del paciente durante las consultas.

Esta fragmentación obliga a los profesionales médicos a solicitar repetidamente la misma información a los pacientes, duplicar exámenes por falta de acceso a resultados previos y dedicar tiempo excesivo a búsquedas manuales de documentación clínica. Todo esto reduce significativamente el tiempo disponible para la atención directa del paciente.

La comunicación médico-paciente fuera del horario de consulta resulta prácticamente inexistente en la mayoría de instituciones. Los pacientes enfrentan dificultades para aclarar dudas sobre tratamientos, reportar efectos secundarios o solicitar renovación de recetas, viéndose obligados a agendar nuevas consultas para resolver inquietudes menores que podrían atenderse mediante comunicación directa.

Esta limitación genera saturación en las agendas médicas, aumenta los costos tanto para pacientes como para instituciones y crea ansiedad en pacientes que requieren orientación oportuna sobre su estado de salud.

**Antecedentes y Evidencia del Problema**Según el estudio *"Digitalización en Salud: Estado Actual en Chile"* realizado por la Asociación Chilena de Informática Médica (ACHIM) en 2023, el 68% de los centros médicos privados aún utilizan sistemas híbridos papel-digital, mientras que solo el 23% cuenta con sistemas completamente digitalizados. Estos datos evidencian un retraso significativo en adopción tecnológica comparado con países desarrollados donde la digitalización supera el 85%.

El informe del Ministerio de Salud sobre *"Eficiencia en Atención Primaria"* documenta que los profesionales médicos destinan en promedio 34% de su tiempo de consulta a tareas administrativas, incluyendo búsqueda de información, registro manual de datos y coordinación de citas. Esta proporción contrasta negativamente con el estándar internacional recomendado de 15%, indicando una pérdida significativa de productividad que podría redistribuirse hacia atención directa del paciente.

La encuesta nacional *"Experiencia del Paciente en Salud Digital"*, aplicada por la Superintendencia de Salud en 2024, revela que el 76% de los pacientes experimenta frustración al no poder acceder fácilmente a sus resultados de exámenes. Además, el 82% considera insuficiente la comunicación con sus médicos tratantes entre consultas, y el 91% estaría dispuesto a utilizar plataformas digitales para gestionar su información de salud si estas garantizarán seguridad y facilidad de uso.

Los datos de saturación hospitalaria proporcionados por el Departamento de Estadísticas e Información de Salud (DEIS) muestran que aproximadamente el 31% de las consultas médicas ambulatorias corresponden a seguimientos rutinarios o resolución de dudas que podrían gestionarse mediante telemedicina o comunicación asíncrona. Esto evidencia una oportunidad significativa de optimización de recursos mediante herramientas tecnológicas apropiadas.

**Origen y Manifestaciones del Problema**La problemática tiene su origen en la adopción tardía de tecnologías de información en el sector salud chileno, agravada por la resistencia al cambio institucional, limitaciones presupuestarias para inversión tecnológica y falta de estándares unificados para sistemas de información sanitaria.

La pandemia de COVID-19 aceleró la conciencia sobre estas limitaciones, exponiendo la fragilidad de sistemas no preparados para atención remota y gestión digital de información clínica.

Las manifestaciones específicas del problema incluyen:

Pérdida de información clínica crítica por falta de sistemas de respaldo adecuados.

Demoras excesivas en acceso a resultados de exámenes que afectan la oportunidad de tratamiento.

Duplicación de procedimientos diagnósticos por falta de acceso a información previa.

Comunicación ineficiente que genera consultas innecesarias y saturación de agendas médicas.

Experiencia del paciente deteriorada por procesos administrativos complejos y acceso limitado a su propia información de salud.

La seguridad de la información constituye una preocupación adicional crítica. Muchas instituciones utilizan sistemas obsoletos vulnerables a brechas de seguridad, almacenan información sensible sin cifrado adecuado y carecen de protocolos robustos de control de acceso. Esto expone datos médicos confidenciales a riesgos de filtración que podrían resultar en sanciones regulatorias y pérdida de confianza del paciente.

**Impacto Económico y Operacional**El impacto económico de esta problemática es significativo. La Cámara de Comercio de Santiago estima que la ineficiencia en procesos administrativos genera sobrecostos operacionales del 23% en centros médicos privados.

La duplicación de exámenes por falta de acceso a información previa representa un gasto adicional promedio de $180.000 por paciente crónico anualmente.

La rotación de personal médico y administrativo también se ve afectada negativamente por la frustración generada por sistemas obsoletos. Según el estudio *"Retención de Talento Médico"* del Colegio Médico de Chile, el 43% de profesionales médicos jóvenes considera la calidad de herramientas tecnológicas como un factor determinante en su decisión de permanencia en una institución.

**Necesidad de Solución Integral**La convergencia de estos factores evidencia la necesidad urgente de una solución tecnológica integral que aborde simultáneamente la gestión eficiente de información clínica, comunicación fluida médico-paciente, optimización de procesos administrativos y seguridad robusta de datos.

La solución debe ser suficientemente flexible para adaptarse a diferentes tipos de instituciones de salud, desde consultorios individuales hasta clínicas de mayor complejidad. Además, debe mantener estándares de seguridad y usabilidad que faciliten su adopción por parte de usuarios con distintos niveles de competencia tecnológica.

La experiencia internacional demuestra que plataformas médicas integradas pueden reducir significativamente los costos operacionales, mejorar la satisfacción del paciente y aumentar la productividad médica.

Países como Estonia y Dinamarca han logrado reducciones del 40% en tiempo administrativo y aumentos del 35% en satisfacción del paciente mediante la implementación de sistemas similares al propuesto por MedInfo. Estos resultados validan la viabilidad y beneficios potenciales de esta aproximación tecnológica en el contexto del sector salud chileno.

## Objetivos del Proyecto (general y específicos)

**Objetivo General**Desarrollar e implementar un sistema integral de gestión médica que optimice la comunicación médico–paciente y la administración de información clínica mediante dos aplicaciones especializadas (un dashboard profesional para médicos y una aplicación móvil para pacientes). El sistema integrará monitoreo biométrico en tiempo real, gestión documental avanzada y comunicación instantánea, bajo estrictos criterios de seguridad ISO 27001, escalabilidad tecnológica y experiencia de usuario diferenciada por roles.

**Objetivos Específicos**

Consolidar la arquitectura tecnológica del sistema MedInfo mediante la finalización del desarrollo del dashboard médico profesional y la integración completa con el backend Flask, garantizando la funcionalidad integral de gestión de archivos médicos, sistema de chat en tiempo real, administración de horarios y citas, seguimiento de exámenes clínicos y comunicación bidireccional médico–paciente, estableciendo las bases para el despliegue productivo y el escalamiento posterior.

Implementar y validar el sistema de seguridad y cumplimiento normativo alineado con ISO 27001, desarrollando mecanismos robustos de autenticación JWT diferenciada por roles, control de acceso granular a recursos médicos, auditoría completa de eventos críticos, cifrado de datos en tránsito y en reposo, y políticas de retención de información clínica, asegurando la confidencialidad, integridad y disponibilidad de los datos sensibles de salud.

Optimizar la experiencia de usuario y la funcionalidad operativa del sistema mediante la implementación de interfaces responsivas e intuitivas, capacidades offline con sincronización automática, notificaciones en tiempo real para eventos médicos críticos, sistema de archivos médicos sin límite de tamaño con almacenamiento híbrido local–nube, y herramientas de comunicación instantánea con capacidad de adjuntos, mejorando significativamente la eficiencia en la gestión clínica y la satisfacción del paciente.

Establecer la infraestructura de producción y procesos de entrega continua mediante el despliegue del sistema en Microsoft Azure App Service (Linux) con capacidad de escalamiento horizontal, implementación de pipeline CI/CD para actualizaciones automáticas, configuración de monitoreo y alertas operacionales, establecimiento de respaldos automatizados y plan de recuperación ante desastres, asegurando una disponibilidad del servicio superior al 99,5% y la capacidad de crecimiento según la demanda institucional.

Preparar la base tecnológica para la integración futura de dispositivos biométricos inteligentes (smartwatches, monitores de presión arterial, glucómetros) mediante el desarrollo de APIs especializadas para recepción y procesamiento de datos fisiológicos en tiempo real, algoritmos de detección de anomalías en parámetros vitales, sistema de alertas automáticas para valores críticos y dashboard de monitoreo continuo que permita a los médicos supervisar remotamente el estado de salud de sus pacientes y tomar decisiones clínicas basadas en datos objetivos y actualizados.

Estos objetivos específicos responden directamente al *“cómo”* del proyecto, estableciendo metas concretas y medibles que abarcan desde la consolidación tecnológica actual hasta la preparación para capacidades avanzadas de telemedicina y monitoreo remoto. Cada objetivo define entregables específicos, criterios de aceptación claros y métricas de éxito que permitirán evaluar el progreso y cumplimiento del proyecto dentro del marco temporal establecido para la Fase 2.

La estructura de estos objetivos reconoce que el sistema MedInfo se encuentra en una etapa de maduración donde las funcionalidades centrales ya están implementadas y validadas, enfocándose ahora en la optimización, seguridad, despliegue productivo y preparación para futuras evoluciones que amplíen significativamente el valor agregado para instituciones de salud y sus pacientes.

## Competencias del Perfil de Egreso

El proyecto MedInfo se encuentra alineado con las competencias del perfil de egreso de Ingeniería en Informática de Duoc UC, ya que integra diversas áreas técnicas y de gestión. Entre las principales competencias aplicadas destacan:

Administrar la configuración de ambientes, servicios de aplicaciones y bases de datos en un entorno empresarial :

Se requiere implementar y mantener servicios de backend, frontend y bases de datos que aseguren la operatividad del sistema de gestión de salud, garantizando continuidad y disponibilidad.

Construir modelos de datos para soportar los requerimientos de la organización de acuerdo a un diseño definido y escalable en el tiempo:

La plataforma exige un modelo de datos sólido que permita manejar información sensible (expedientes clínicos, diagnósticos, recetas, monitoreo biométrico) de manera eficiente, segura y escalable.

Realizar pruebas de calidad tanto de los productos como de los procesos utilizando buenas prácticas definidas por la industria:

Dado que se trata de un sistema crítico de salud, es indispensable garantizar la fiabilidad mediante pruebas funcionales, de seguridad, rendimiento y usabilidad.

Gestionar proyectos informáticos, ofreciendo alternativas para la toma de decisiones de acuerdo con los requerimientos de la organización:

Se aplicará la metodología ágil Scrum, permitiendo una gestión eficiente del tiempo, recursos y entregables del proyecto.

Construir el modelo arquitectónico de una solución sistémica que soporte los procesos de negocio de acuerdo a requerimientos y estándares industriales:

Se diseñará una arquitectura modular y escalable, integrando aplicaciones móviles, web y servicios backend de forma coherente y mantenible.

## Asignación de roles

Cada integrante del equipo asume un rol específico en función de sus fortalezas técnicas y de gestión, asegurando una adecuada distribución de responsabilidades:

| **Integrante** | **Rol Técnico Específico** | **Responsabilidades Clave en la FASE 2** | **Propósito (Justificación del Aporte)** |
| --- | --- | --- | --- |
| **Mauricio Guzmán** | **Arquitecto Backend y DevOps** | **Liderazgo en el *setup* del ambiente de producción: Configuración de la Base de Datos Neon PostgreSQL y desarrollo de los Scripts de *Deployment*. Implementación inicial de la API REST de Usuarios (CRUD).** | **Asegurar la disponibilidad y el *deployment* automatizado del Back-end.** |
| **Matías Rodríguez** | **Lead Frontend Developer (Angular/Ionic)** | **Definición de la Arquitectura del *Monorepo*. Diseño de la experiencia de usuario (UX) de las dos interfaces y desarrollo de la capa de servicios para el consumo de la API.** | **Garantizar la usabilidad y la coherencia del diseño en ambas plataformas (Dashboard y App Móvil).** |
| **Rafael Diaz** | **Ingeniero de Datos y Seguridad (ISO 27001)** | **Diseño y migración del Modelo Entidad-Relación (E-R). Implementación de los *middlewares* de autenticación JWT y el control de acceso basado en roles para cumplir con ISO 27001.** | **Establecer la integridad de los datos y la seguridad desde la capa de la API.** |

## Metodología utilizada en el Proyecto

El desarrollo y la implementación del proyecto MedInfo se llevarán a cabo utilizando la metodología ágil **Scrum**. Este enfoque iterativo e incremental fue seleccionado tras evaluar distintas alternativas de gestión de proyectos, determinando que es el ideal para un sistema de alta complejidad técnica y requisitos de seguridad críticos (ISO 27001).

**Evaluación de Metodologías Alternativas**

Para fundamentar la elección de Scrum, se analizaron otros enfoques comunes en el desarrollo de software y se descartaron por las siguientes razones:

**Metodología en Cascada (Waterfall):**

*Enfoque:* Lineal y secuencial (Requisitos -> Diseño -> Desarrollo -> Pruebas).

*Razón del descarte:* El modelo en Cascada exige una definición total de los requisitos al inicio. Dado que MedInfo es un proyecto de innovación que integra tecnologías biométricas y APIs externas (cuyas especificaciones pueden cambiar), la rigidez de Cascada representaba un riesgo alto. Un error en la fase de diseño inicial podría haber comprometido todo el desarrollo sin posibilidad de corrección temprana.

**Kanban:**

*Enfoque:* Flujo continuo de trabajo visualizado en tableros, sin iteraciones de tiempo fijo.

*Razón del descarte:* Aunque Kanban es excelente para mantenimiento, carece de los "time-boxes" (ciclos de tiempo fijo) estrictos que ofrece Scrum. Para un proyecto de tesis con plazos académicos improrrogables, la estructura de *Sprints* de Scrum provee la presión necesaria para asegurar entregas parciales y medibles cada dos semanas, evitando la dispersión del alcance.

**Implementación de Scrum en MedInfo**

Scrum estructura el trabajo en ciclos de corta duración llamados **Sprints**, permitiendo que el equipo gestione el riesgo y valide la funcionalidad de manera periódica.

Para la Fase 2 (Diseño e Implementación), el equipo estructuró el trabajo en dos Sprints de dos semanas de duración cada uno. Esta división se realizó priorizando la infraestructura sobre la funcionalidad final:

**Sprint 1 (Base Arquitectónica):** Se enfocó en establecer la columna vertebral técnica. Los objetivos clave fueron la configuración y migración a Azure Database for PostgreSQL, el setup del Backend Flask, y la implementación del middleware de autenticación JWT para la capa de seguridad.

**Sprint 2 (Modelado y Pruebas de Integración):** Se dedicó al desarrollo de los Modelos Entidad-Relación (E-R) y a la integración de Frontends. Se resolvieron los desafíos de comunicación CORS entre los entornos de desarrollo (Angular/Flask) y se realizaron las primeras pruebas de login/registro.

**Fases del Ciclo Scrum**

El proceso se define a través de las siguientes etapas principales, las cuales se repiten en cada Sprint:

**Planificación del Sprint (Sprint Planning):**

*Función:* Definir el "Qué" y el "Cómo". El equipo selecciona tareas de alta prioridad del *Product Backlog* y estima el esfuerzo técnico requerido. El resultado es el *Sprint Backlog*.

**Ejecución y Reunión Diaria (Daily Scrum):**

*Función:* Asegurar la transparencia y coordinación. Reunión diaria de 15 minutos donde se sincroniza el progreso del Backend y Frontend, identificando bloqueos técnicos (ej. problemas con dependencias o APIs).

**Revisión del Sprint (Sprint Review):**

*Función:* Inspeccionar el incremento. Al finalizar el Sprint, se presenta la funcionalidad terminada (ej. módulo de autenticación JWT operativo) para validar que cumple con los estándares de seguridad y usabilidad.

**Retrospectiva del Sprint (Sprint Retrospective):**

*Función:* Mejora continua. El equipo analiza qué salió bien y qué procesos técnicos u organizativos se deben optimizar para el siguiente ciclo.

### 

**Justificación de la elección de Scrum**

En conclusión, se seleccionó Scrum por su capacidad de adaptación frente a la rigidez de modelos tradicionales como Cascada. El desarrollo de MedInfo implica alta complejidad técnica e incertidumbre (integraciones biométricas y seguridad de datos), lo que hace indispensable un marco de trabajo que permita:

**Adaptabilidad:** Iterar y ajustar rápidamente el producto frente a nuevos hallazgos técnicos o requerimientos clínicos, algo imposible en modelos lineales.

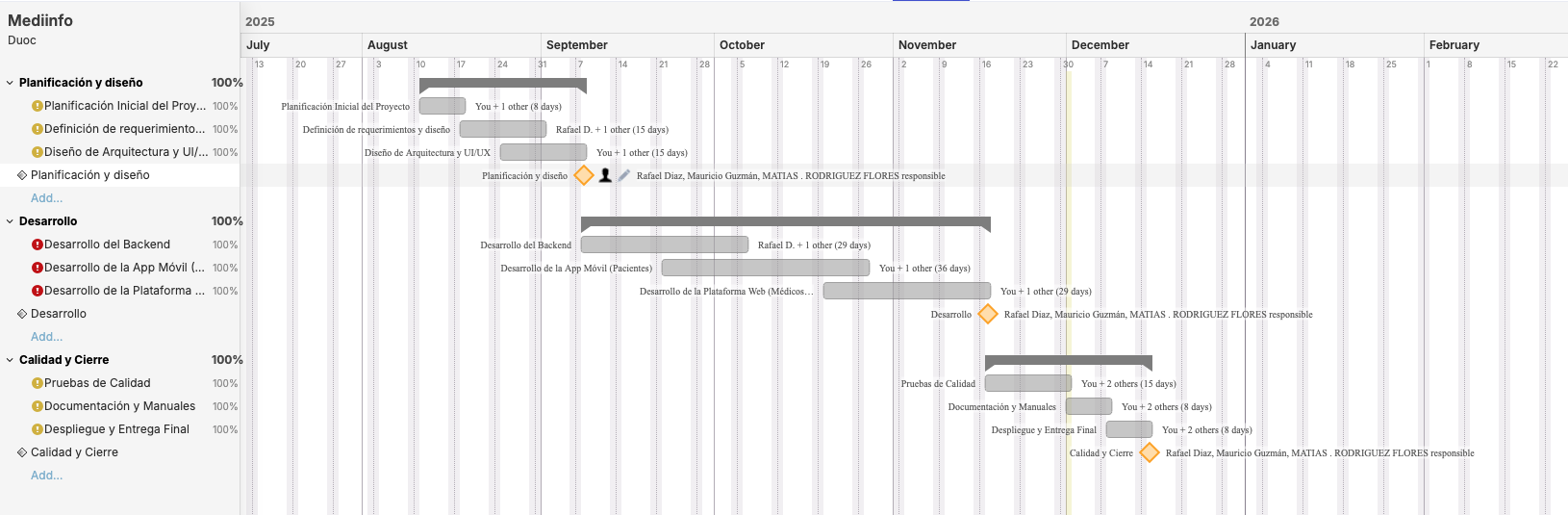
**Control de Riesgos:** Detectar fallos de seguridad o arquitectura en etapas tempranas gracias a las revisiones al final de cada Sprint.

**Entrega de Valor Incremental:** A diferencia de Kanban, los Sprints obligan a tener entregables tangibles (como la autenticación segura o el chatbot) en fechas fijas, alineándose con el cronograma académico de la tesis.

**Calidad Continúa:** La validación constante asegura el cumplimiento normativo (ISO 27001) durante el desarrollo, en lugar de dejar la auditoría de calidad solo para el final del proyecto.

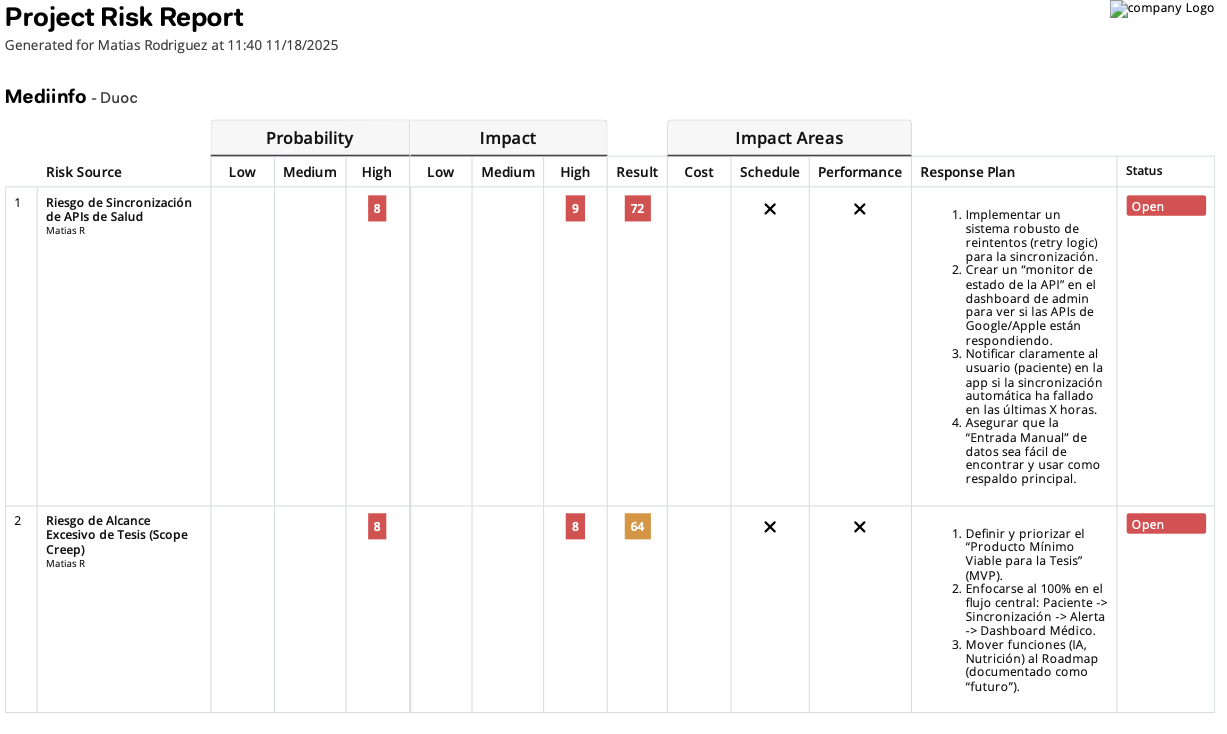
Por estas razones, Scrum es la metodología más adecuada para garantizar la eficiencia, control y éxito del proyecto MedInfo.

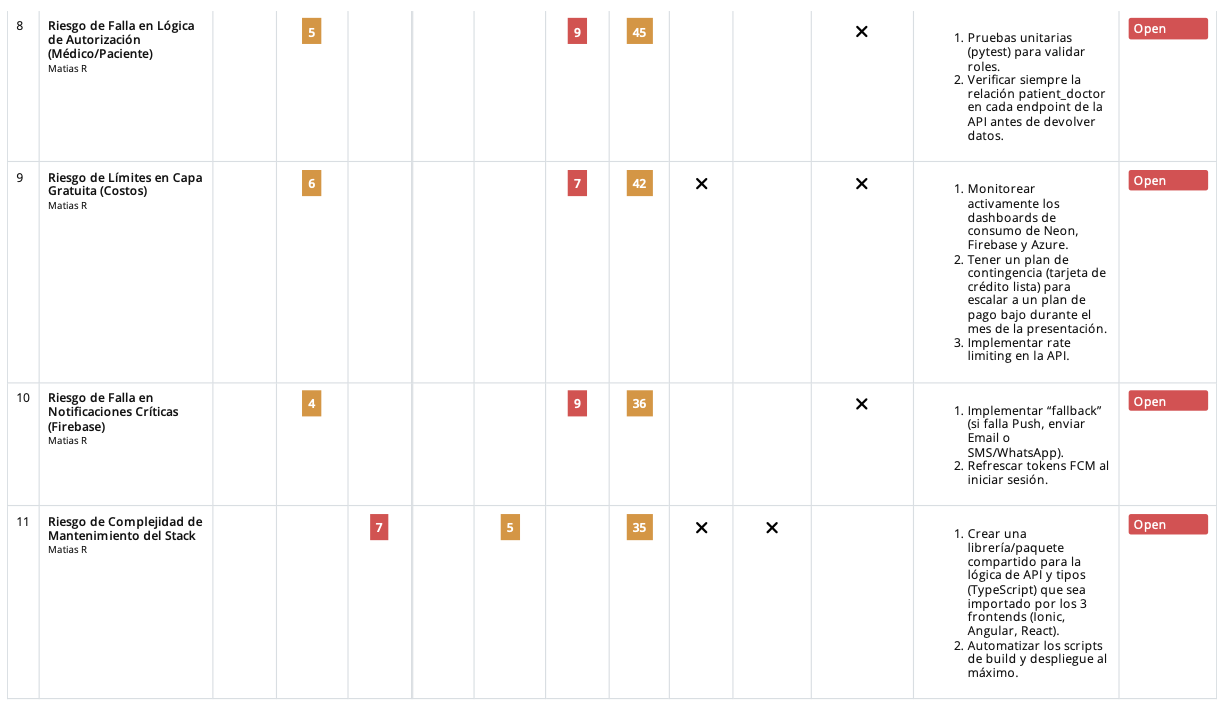
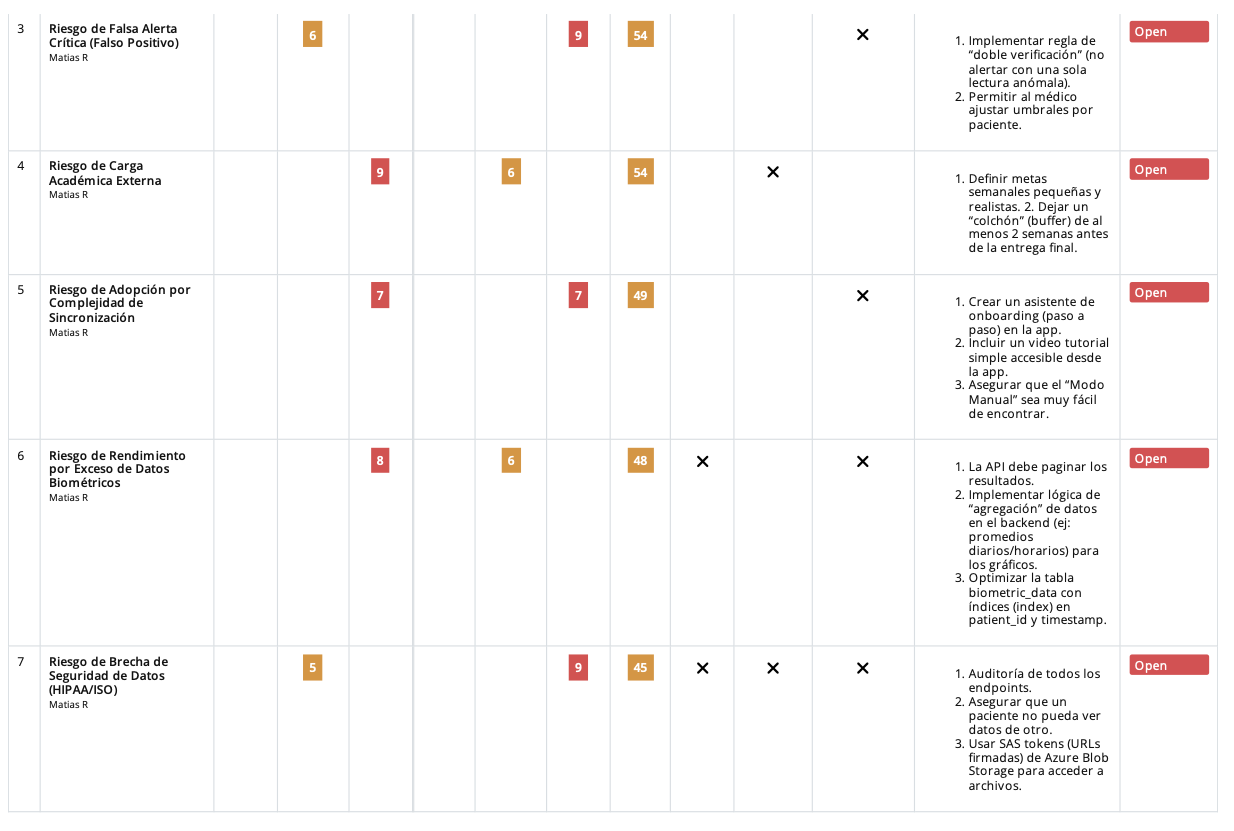
## Creación de cronograma asociado al Proyecto (Carta Gantt)



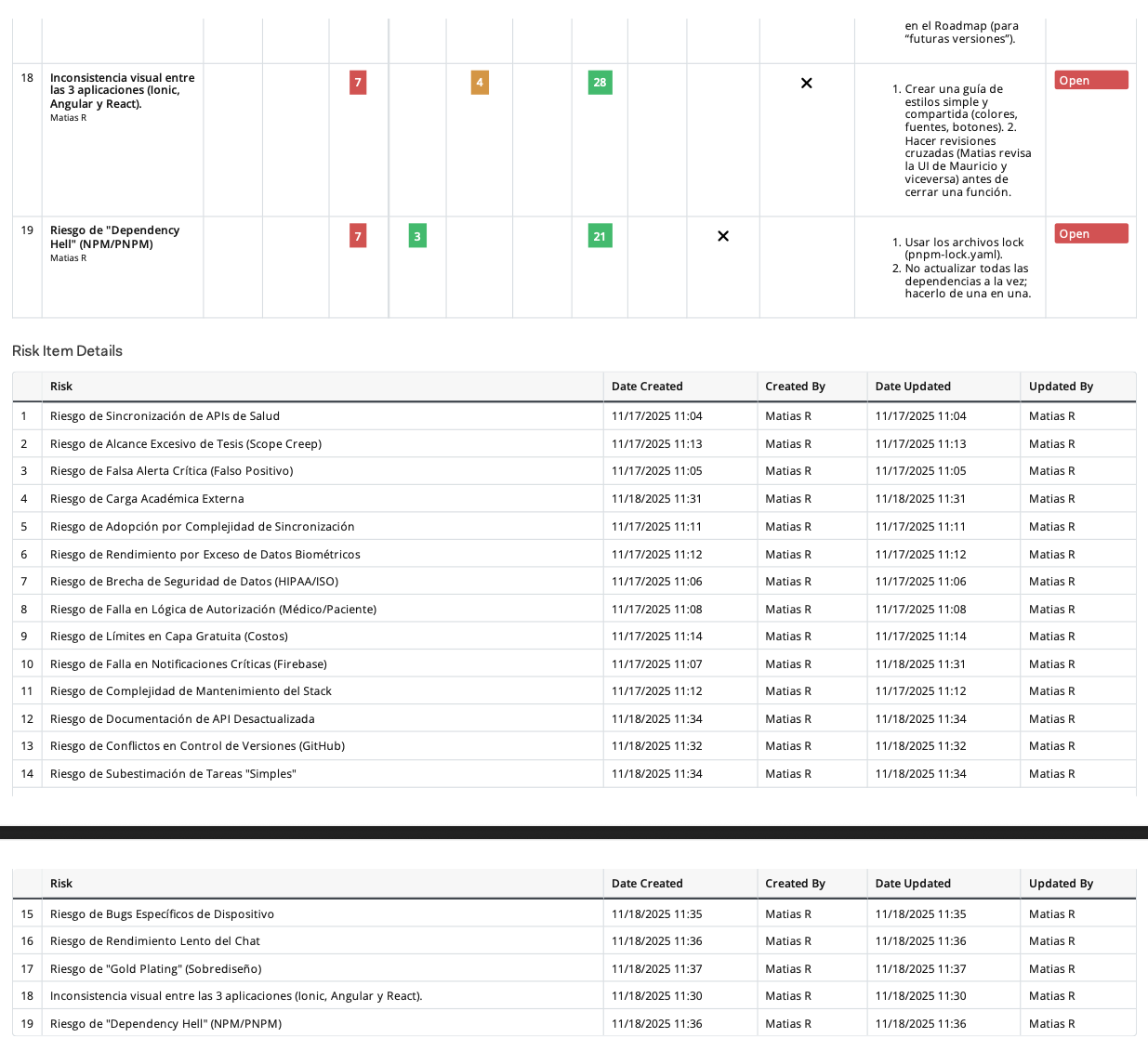
## Riesgos Asociados al Proyecto

La gestión de riesgos es un proceso vital en el desarrollo de MedInfo para anticipar y mitigar las amenazas que podrían comprometer la calidad, seguridad (ISO 27001) y el cronograma. A continuación, se presenta la matriz de riesgos identificados durante la Fase 2 (Implementación), incluyendo el plan de mitigación.





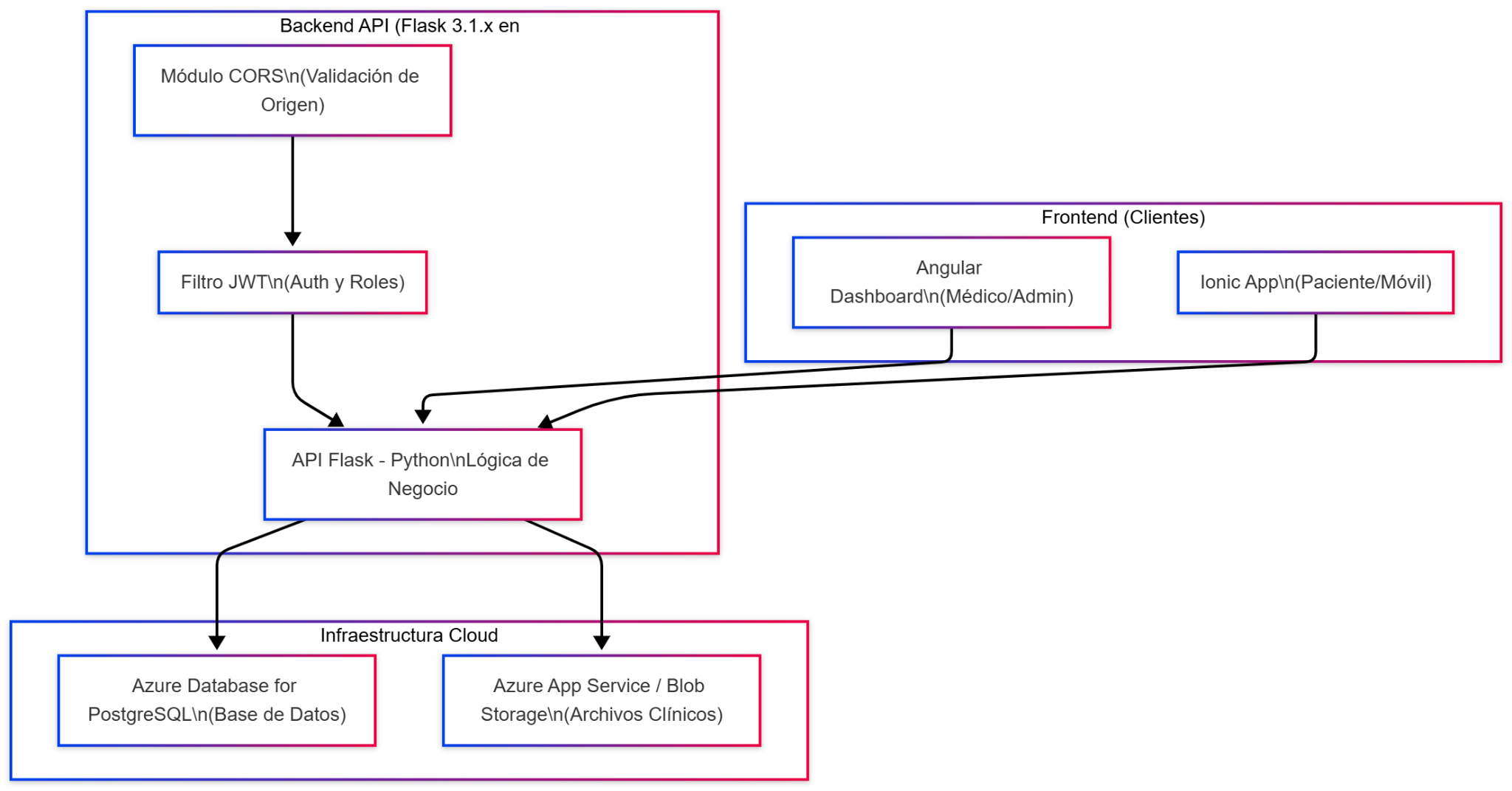




## Implementación del Proyecto

### **Diseño y arquitectura de la solución**

El proyecto MedInfo se implementó bajo un modelo de **Arquitectura Lógica de Tres Capas** con un enfoque modular y desacoplado, priorizando la **seguridad (ISO 27001)**, la escalabilidad y la gestión separada de la interfaz de usuario. Este diseño sigue el principio de **separación de responsabilidades**, asegurando que el *backend* funcione como una capa de servicio centralizado para ambas plataformas de usuario. La estructura completa se ilustra en el **Anexo B:** [**Diagrama de Arquitectura Lógica**.](https://drive.google.com/file/d/1TJdjjEltDLLb02yLG3djZGozKHfIgDTg/view?usp=drive_link) **Anexo B**



**Capa de Presentación (Frontend Clientes)**

Esta capa incluye dos aplicaciones desacopladas que acceden al *Backend* exclusivamente a través de peticiones seguras **HTTPS (REST)** y **WebSockets** (para la funcionalidad de chat), garantizando la unicidad del acceso.

**Dashboard Web (Angular y React):** Interfaz para **Médicos y Administradores**, enfocada en la gestión de datos complejos y reportes clínicos avanzados.

**App Móvil (Ionic):** Aplicación móvil para **pacientes**, optimizada para el acceso a citas, historiales y la captura de datos biométricos en tiempo real.

Capa de Aplicación (Backend API Flask)

La lógica de negocio se centraliza en un **API RESTful** desarrollada con **Flask (Python)**. Es el único punto que tiene acceso directo a la Base de Datos.

**Seguridad y Flujo de Control:**

**Filtro JWT (JSON Web Tokens):** Este middleware valida la autenticidad y el rol del usuario en cada petición, negando automáticamente el acceso si el token es inválido o está ausente.

**Módulo CORS:** Asegura que solo las aplicaciones *Frontend* autorizadas (Dashboard y App Móvil) puedan interactuar con la API, previniendo ataques de falsificación de peticiones.

**Gestión de Datos No Estructurados:**

El Backend gestiona el almacenamiento de archivos grandes (ej. radiografías, informes PDF) que serán depositados en un servicio externo como Azure Blob Storage/App Service (Linux) (evitando saturar la base de datos relacional).

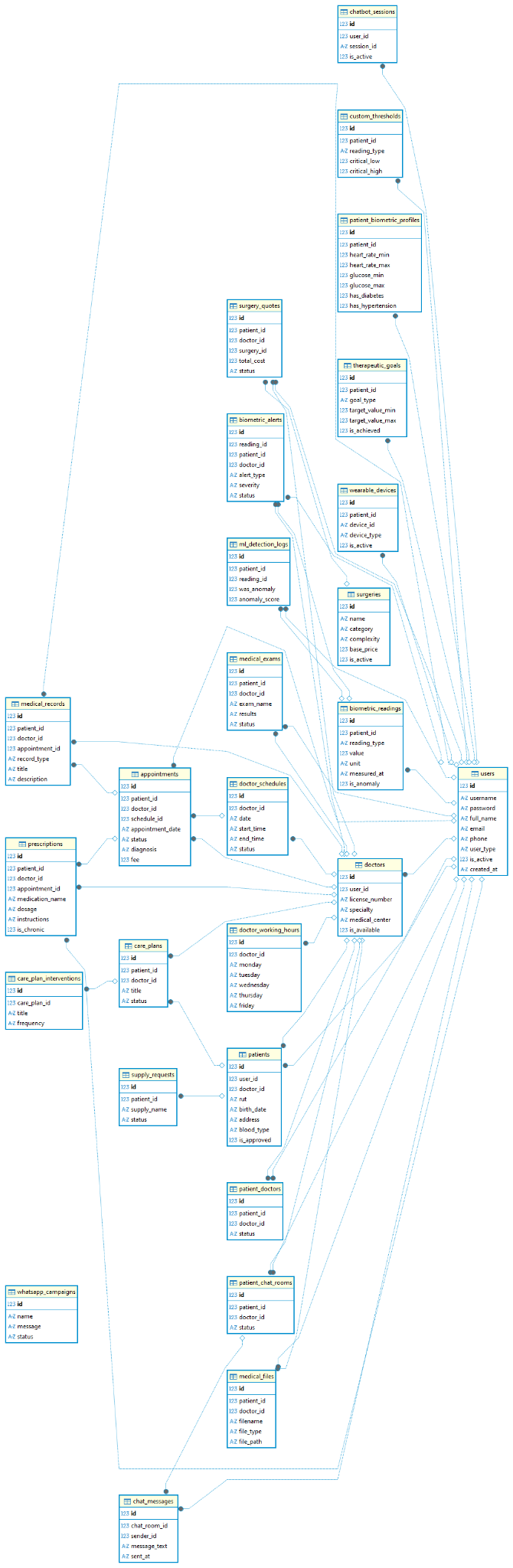
**Capa de Datos (Azure Database for PostgreSQL)**

La persistencia de datos relacionales se gestiona mediante Azure Database for PostgreSQL. Esta solución serverless y de código abierto garantiza alta disponibilidad y escalabilidad horizontal.

**Justificación:** Se eligió por su **integridad de datos**, seguridad transaccional, y su capacidad de **escalar automáticamente** para manejar crecimientos exponenciales de datos clínicos sin interrupción, mitigando el **Riesgo R3 (Problemas de Escalabilidad de BD)**.

El diseño detallado de las tablas y sus relaciones se encuentra en el **Anexo A:** [**Diagrama Entidad-Relación (DER)**](https://drive.google.com/file/d/1tonSI-72kVOq9UYox2F7qHDNXrOWJQhr/view?usp=drive_link).

Anexo A



**Modelo de Datos Clave para la Escalabilidad**

El modelo de datos relacional se diseñó para soportar la información sensible y las relaciones críticas entre los roles principales. La escalabilidad se garantiza mediante una clara separación de las entidades transaccionales y las entidades de historial, minimizando la latencia en las consultas diarias.

Las entidades clave que soportan los requerimientos de MedInfo son:

**USUARIO (Entidad Core):** Define el rol de la persona (Médico, Paciente). Contiene datos de autenticación y seguridad (**JWT**).

**PACIENTE:** Contiene datos demográficos y de contacto. Se enlaza a la tabla **USUARIO** mediante una **Clave Foránea (FK)**.

**MÉDICO:** Contiene la especialidad y número de licencia. Se enlaza a la tabla **USUARIO**.

**HISTORIAL CLÍNICO:** Entidad principal para la confidencialidad. Contiene diagnósticos, notas de evolución y referencias a los archivos. Tiene FKs a **PACIENTE** y **MÉDICO** (para el médico que registra la nota).

**MONITOREO BIOMÉTRICO:** Entidad diseñada para la escalabilidad de tiempo. Almacena series de tiempo de datos biométricos (ej. presión, glucosa). Está optimizada para consultas rápidas por rango de fecha.

**CITA MEDICA:** Entidad transaccional. Contiene la hora, fecha, estado (Pendiente, Confirmada, Cancelada) y FKs a **PACIENTE** y **MÉDICO**.

### **Requerimientos técnicos**

Se detalla el listado de dispositivos, insumos y herramientas esenciales para la implementación, asegurando la operatividad, seguridad y escalabilidad del proyecto MedInfo.

| Categoría | Recurso / Herramienta | Versión / Descripción | Justificación en el Proyecto |
| --- | --- | --- | --- |
| Backend / API | Python / Flask | 3.11+ / Última estable | Desarrollo de la lógica de negocio y APIs **REST** seguras. |
| Base de datos (Dev) | PostgreSQL / Neon | **PostgreSQL 16.x** (Última estable) | Base de datos relacional segura (soporte para **cifrado en reposo**). |
| Autenticación | PyJWT | Última estable | Implementación de la autenticación por **JSON Web Tokens (JWT)**. |
| Frontend Médico | Angular | **17.x** | *Framework* para el **Dashboard Médico**, ideal para interfaces ricas en datos y módulos grandes. |
| Frontend Paciente | Ionic / Capacitor | **7.x** | App Móvil Híbrida para iOS/Android con acceso a funcionalidades nativas y notificaciones. |
| Infraestructura Cloud | Azure App Service / Storage | App Service, Blob Storage | Alojamiento del Backend y almacenamiento seguro de archivos clínicos grandes (imágenes, PDFs). |
| Herramientas de Desarrollo | Git & GitHub | Plataforma cloud | Control de versiones y soporte a procesos de **CI/CD** simulados. |
| Pruebas API | Postman | Última estable | Herramienta esencial para la prueba y documentación de los *endpoints* del Backend (Flask). |

### 

### 

### 

### 

### 

### 

### 

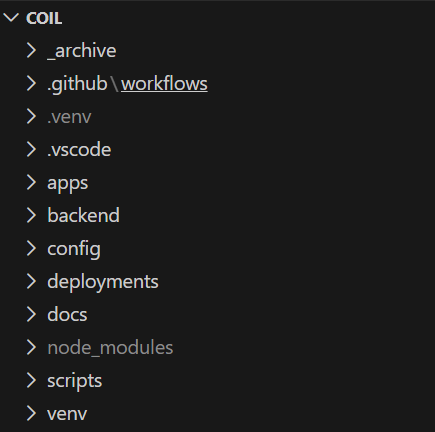
### **Desarrollo de la solución**

El desarrollo de MedInfo en la Fase 2 se enfocó en consolidar la arquitectura de software sobre el modelo de **Monorepo**, una técnica de sistematización que permite unificar los repositorios del Backend (Flask) y los Frontends (Angular/Ionic/React).

**Sistematización del Desarrollo (Monorepo y Git)**

La utilización de un Monorepo junto a la metodología Git asegura la **trazabilidad** y el **mantenimiento** del código, cumpliendo con los estándares de la disciplina.

**1: Estructura de Proyecto Unificada:** Se organizó el proyecto en carpetas modulares (backend/, apps/, config/) para que cada componente se desarrolle y se versiones de forma aislada, pero bajo un mismo repositorio.

****

**2: Control de Versiones Colaborativo:** El equipo utilizó git para gestionar los *commits*, ramas y *pull requests*, evidenciando el trabajo en equipo y el aporte individual. 

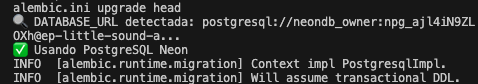
**Implementación del Backend (Flask y SQLAlchemy)**

La implementación del Backend (Flask) se estructuró para mantener la lógica de negocio desacoplada de la capa de datos.

**Paso 3: Mapeo de Modelos con ORM**: Se definió la estructura de las entidades clave (ej. users, patients) en el archivo models.py utilizando SQLAlchemy. Esto asegura la integridad de los datos relacionales en Azure Database for PostgreSQL.



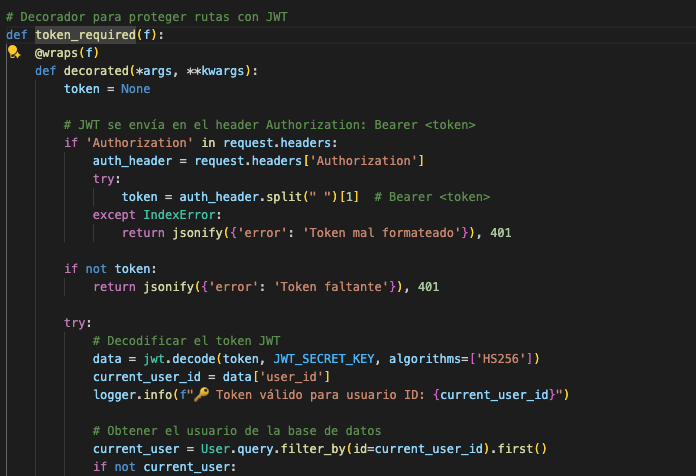
**Paso 4: Migración Controlada con Alembic**: Se utilizó Alembic para generar y aplicar las migraciones (alembic upgrade head) a la base de datos serverless Azure Database for PostgreSQL, asegurando que los cambios en el Modelo E-R sean trazables y reversibles (Técnica de Mantenimiento).



**Implementación de Seguridad (JWT y ISO 27001)**

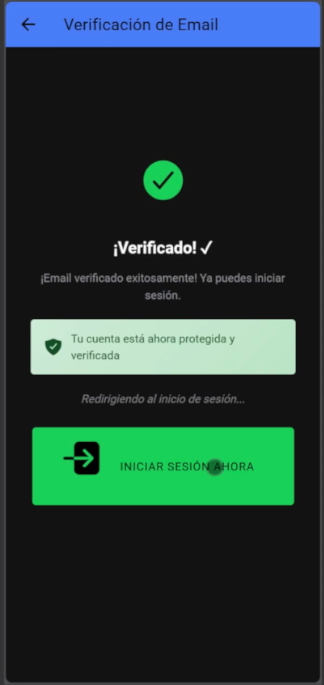
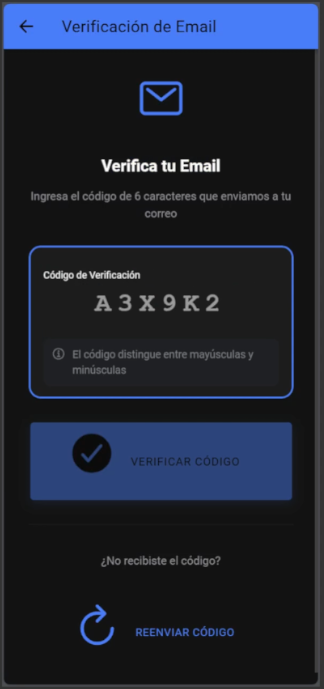
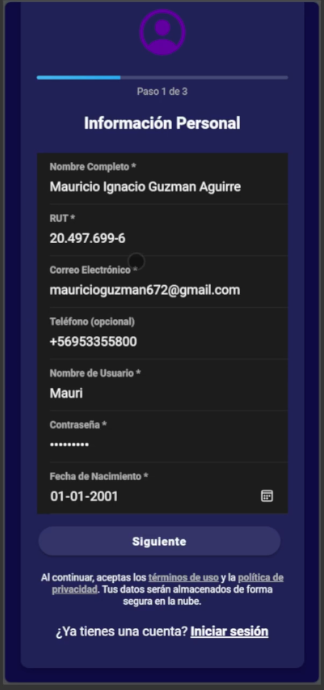
El módulo de seguridad es crítico para el cumplimiento de **ISO 27001**, ya que controla el acceso a los datos médicos.

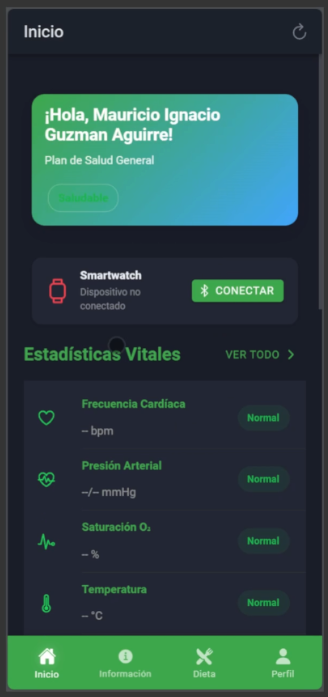
* **Paso 5: Configuración del Filtro JWT:** Se implementó un *middleware* de autenticación que intercepta cada petición a rutas protegidas. Si el *token* **JWT** no está presente o es inválido, el acceso es denegado.

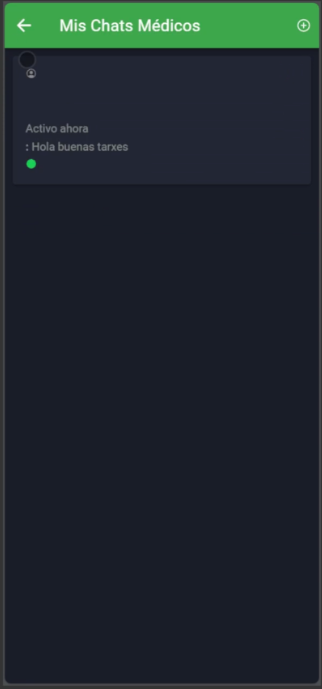
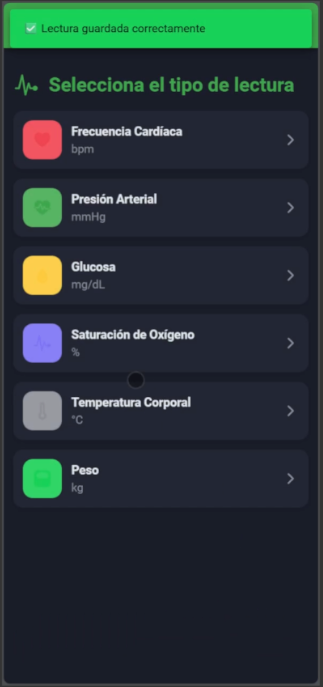


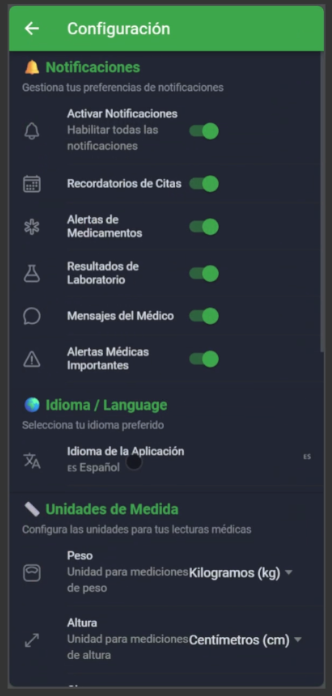
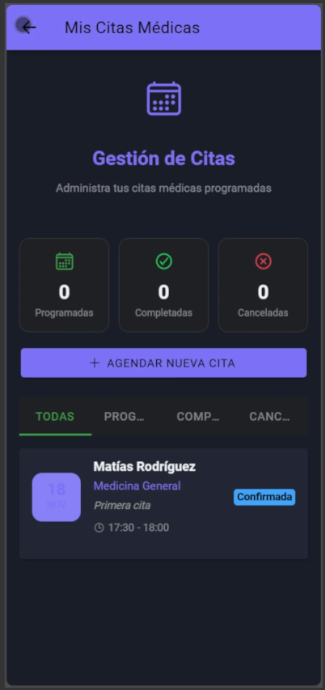
### **Resultados de la solución**

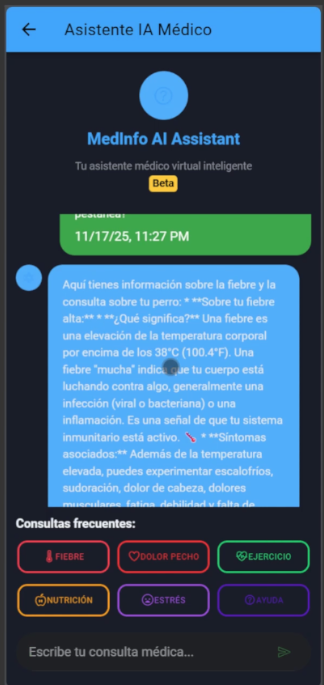
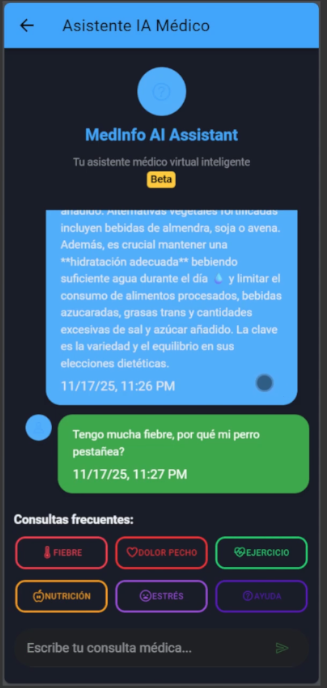
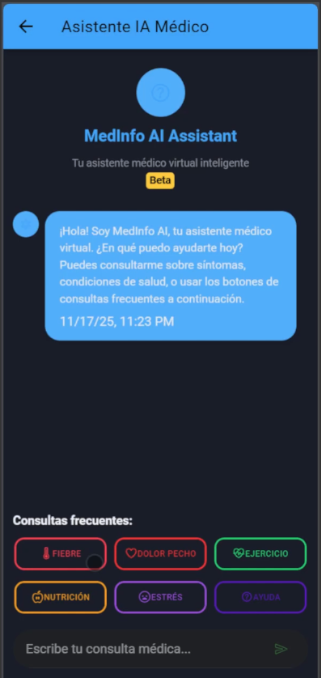
Vistas de la aplicación mobile



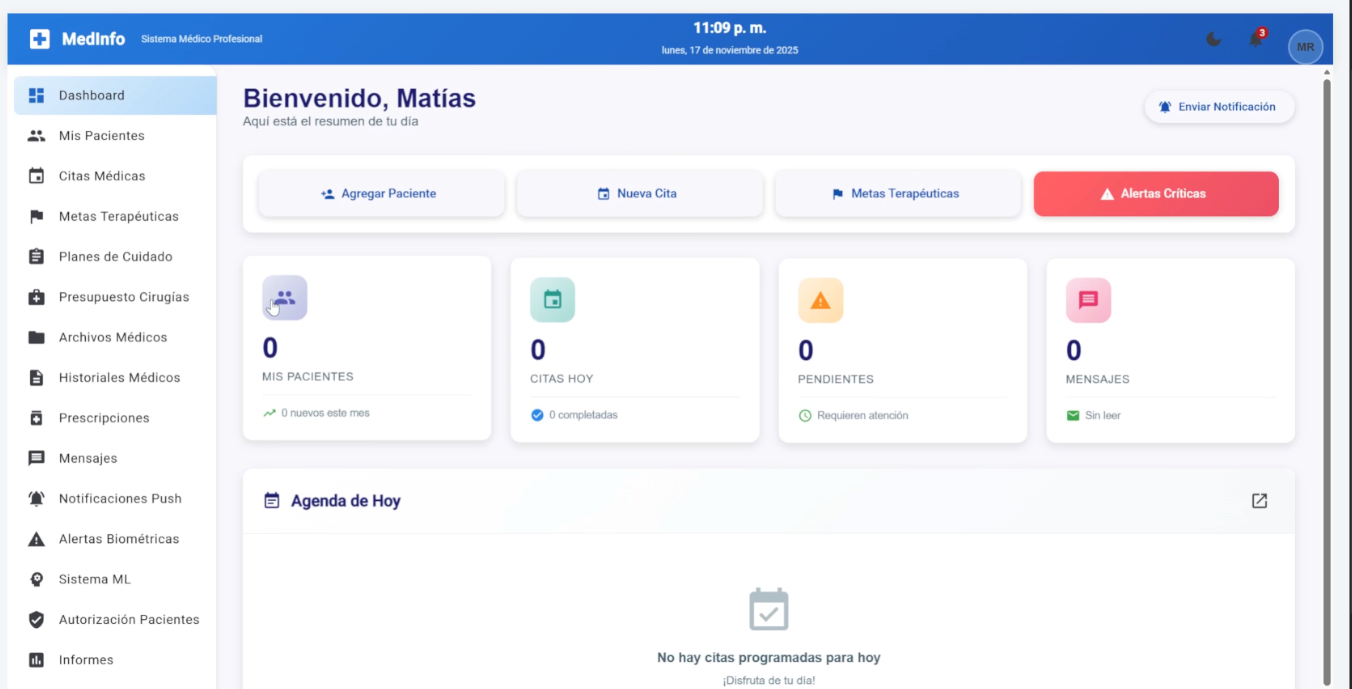


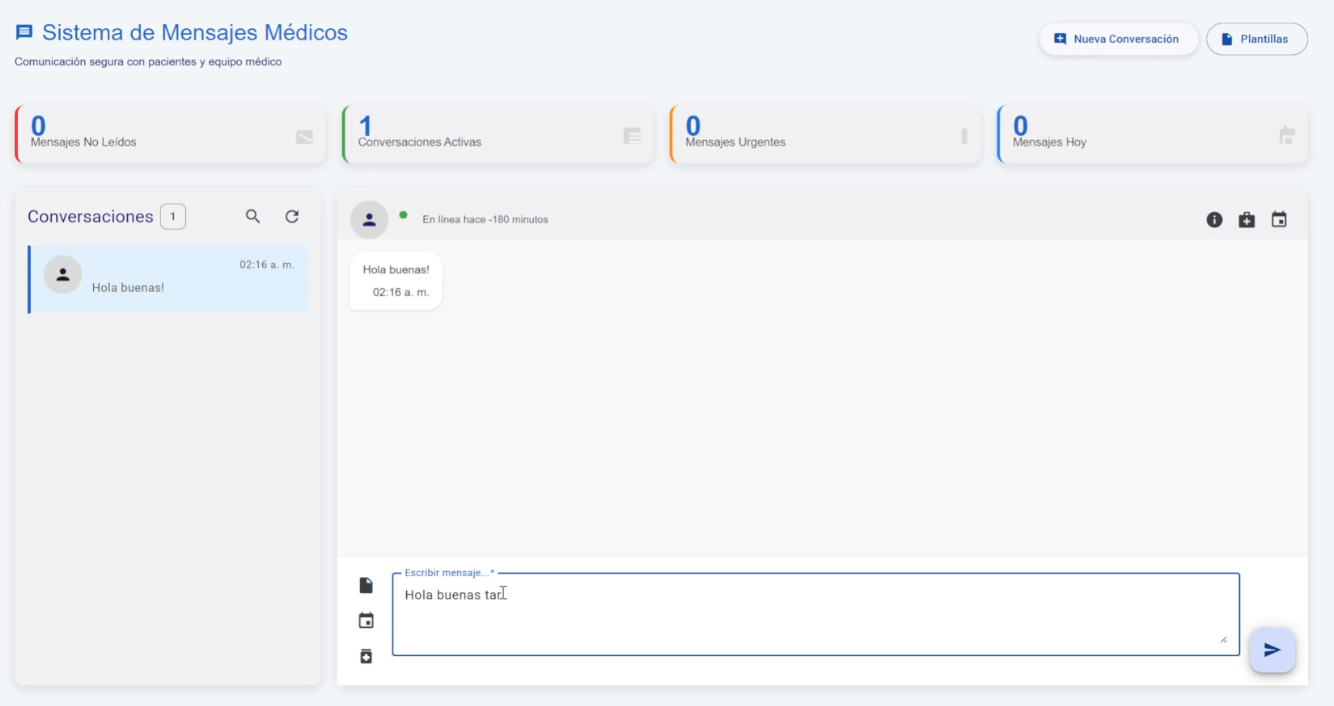
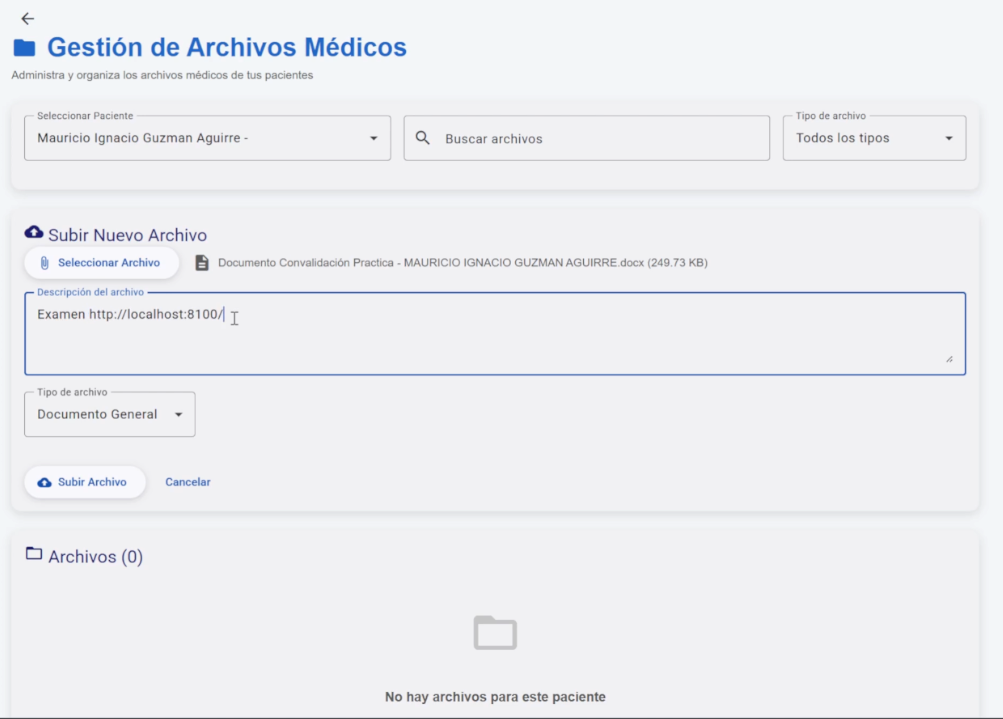






**Vistas Dashboard Médico**





**Resultados de Validación**

Los resultados de la Fase 2 se validaron mediante pruebas funcionales de los componentes clave de la arquitectura, que son la **Autenticación (Seguridad)** y la **Conectividad (Integración)**.

| ID Prueba | Título Formal (Evidencia a Insertar) | Petición de Postman | Código Clave | Lo que Demuestra en el Informe | Evidencia |
| --- | --- | --- | --- | --- | --- |
| P1 | Prueba de Autenticación Exitosa | POST /api/auth/login | 200 OK | Confirma que la conexión es correcta y el *backend* emite el **Token JWT**. | [Anexo C.1](http://anexos/Anexo_C.1.png) |
| P2 | Acceso Autorizado (Uso del JWT) | GET /api/pacientes (con Token válido) | 200 OK | Demuestra que el *middleware* JWT acepta el token válido y la API entrega datos. | [Anexo C.2](http://anexos/Anexo_C.2.png) |
| P3 | Denegación de Acceso (Error 401) | GET /api/pacientes (con Token inválido) | 401 Unauthorized. | **Evidencia de Seguridad** clave. Demuestra que el filtro JWT rechaza peticiones no autorizadas. | [Anexo C.3](http://anexos/Anexo_C.3.png) |
| P4 | Creación de Recurso (Registro en DB Neon) | POST /api/registro o POST /api/usuarios | 201 Created | Demuestra que la API puede escribir nuevos datos de manera segura en la base de datos de **Neon PostgreSQL**. | [Anexo C.4](http://anexos/Anexo_C.4.png) |

## Definición de Recursos y Costos asociados al Proyecto

**Costo de Recursos Humanos (Fase de Desarrollo e Implementación)**

| Rol | Integrante | Horas Estimadas Totales (15 horas/sem×18 sem) | Costo Estimado por Hora (CLP Simulados) | Costo Total por Rol (CLP Simulados) |
| --- | --- | --- | --- | --- |
| Bases de Datos  / Arquitectura | Rafael Díaz | 270 horas | $12.000 | $3.240.000 |
| Backend / App Móvil | Mauricio Guzmán | 270 horas | $11.000 | $2.970.000 |
| Frontend/ QA | Matías Rodríguez | 270 horas | $11.000 | $2.970.000 |
| TOTAL ESTIMADO RECURSOS HUMANOS |  | 810 horas |  | $9.180.000 |

Este costo simula el valor total de las horas de trabajo profesional dedicadas a la construcción del Producto Mínimo Viable (MVP) de MedInfo, siendo el principal rubro de inversión.

**Costo de Transporte (Estimación)**

| Viajes Totales | 8 reuniones (una cada dos semanas) | 8 viajes |
| --- | --- | --- |
| Costo por Viaje (ida y vuelta) | Estimación por vehículo/equipo (combustible, peajes) | $25.000 |
| Costo Total Transporte | 8 viajes x $25.000 | $200.000 |

**Costos Tecnológicos y Operacionales (Estimación Mensual)**

Esta tabla proyecta los costos de infraestructura necesarios para el despliegue de MedInfo en un entorno productivo basado en la nube, garantizando escalabilidad y alta disponibilidad, alineado con la arquitectura definida en 1.10

| Recurso | Proveedor | Justificación | Costo Estimado Mensual (CLP) |
| --- | --- | --- | --- |
| Hosting Web / Backend | Azure App Service (B1) | Alojamiento gestionado para la API Flask, escalable y seguro | $32.000 |
| Base de Datos (DBaaS) | Azure Database for PostgreSQL | Servicio gestionado, backups automáticos, alta disponibilidad y escalabilidad. | $30.000 |
| Almacenamiento de Archivos | Azure Blob Storage (50 GB) | Almacenamiento seguro y redundante para archivos clínicos (imágenes, PDFs). | $10.000 |
| Dominio y Certificado | Azure / Namecheap / SSL | Dominio profesional y certificado HTTPS para cifrado en tránsito. | $7.000 |
| Panel Admin / Dashboard | Azure Static Web Apps | Hosting para panel admin (React) y dashboard médico (Angular). | $3.000 |
| TOTAL ESTIMADO RECURSOS TECNOLÓGICOS |  |  | $82.000 |

La factibilidad económica operacional del sistema MedInfo se establece en **$82.000 CLP mensuales** por infraestructura básica, demostrando un bajo costo operativo una vez finalizada la fase de desarrollo.

**Costo de Viáticos / Alimentación (Estimación)**

| Viático Diario por Persona | Desayuno y Almuerzo | $10.000 |
| --- | --- | --- |
| Costo por Reunión (3 personas) | 3 personas $10.000 | $30.000 |
| Costo Total Viáticos | 8 reuniones $30.000 | $240.000 |

**Costo Total del Proyecto (Sin Utilidad)**

| Costo Total de Recursos Humanos (RR.HH.) | $9.180.000 |
| --- | --- |
| Costo Tecnológico (4.5 meses) | $369.000 |
| Costo de Viáticos | $240.000 |
| Costo de Transporte | $200.000 |
| COSTO TOTAL DEL PROYECTO (A) | $9.989.000 |

**Cálculo de la Utilidad (20%)**

Aplicamos el margen de utilidad del 20% sobre el Costo Total del Proyecto (A).

Utilidad de (20%) ($9.989.000 \* 0.20) = $1.997.800

Sumamos el costo total del proyecto más la utilidad calculada para obtener el nuevo precio de venta.

Precio de Venta = $9.989.000 + $1.997.800 = $11.986.800

| **Costo Total del Proyecto (MVP)** | **$9.989.000** |
| --- | --- |
| **Utilidad (20%)** | **$1.997.800** |
| **PRECIO DE VENTA ESTIMADO** | **$11.986.800** |

# CONCLUSIÓN

El proyecto MedInfo ha completado exitosamente la Fase 2: Diseño e Implementación, estableciendo una **base tecnológica robusta y segura** que aborda directamente la necesidad de modernización y unificación de sistemas en el sector salud. Los objetivos de la fase se han cumplido con rigor, sentando las bases para el éxito de la entrega final.

Los logros clave incluyen:

**Arquitectura Sólida y Segura**: Se ha definido una arquitectura desacoplada (Angular, Ionic, Flask) soportada por PostgreSQL/Azure Database for PostgreSQL, asegurando escalabilidad y disponibilidad. La implementación temprana del control de acceso mediante JWT y la planificación de cifrado en reposo garantizan el cumplimiento de los estándares de seguridad (ISO 27001) contra la filtración de datos.

**Modelo de Datos Coherente:** El diseño del Modelo Entidad-Relación soporta de manera eficiente los datos clínicos, las transacciones de citas y, crucialmente, la serie de tiempo del **monitoreo biométrico**, lo que facilita el desarrollo de futuras funcionalidades de análisis.

**Gestión de Riesgos Proactiva:** La identificación y mitigación de riesgos críticos, como la **Brecha de Seguridad (R1)** y la **Desviación del Alcance (R6)**, demuestran la madurez del equipo para manejar las complejidades del proyecto.

**Planificación Concreta:** La Carta Gantt de 18 semanas, alineada con la metodología **Scrum**, organiza el desarrollo y asegura que las fases de **Pruebas de Calidad** y **Cierre (S15-S18)** dispongan del tiempo necesario para una entrega sin fallas.

El sistema MedInfo se encuentra ahora en el umbral de su despliegue, con el código listo para integrar los componentes finales y comenzar las pruebas formales. La inversión de **$9.180.000 CLP** en desarrollo, con un costo operacional bajo, validó la propuesta de valor del proyecto como una solución eficiente y sostenible para la gestión clínica moderna.

**Individual Conclusions**

The successful completion of Phase 2 has provided significant technical learning experiences to each team member. Below, each individual reflects on their primary contribution and the skills developed during the implementation of the MedInfo architecture (ISO 27001).

**Mauricio Guzmán (Backend Architect & DevOps)**

During this phase, my focus was on establishing the core infrastructure. I successfully integrated the Azure Database for PostgreSQL database and developed the deployment scripts, ensuring the backend is scalable and reliable. The main learning was mastering the serverless architecture for data integrity, which is a critical skill for modern cloud engineering.

**Matías Rodríguez (Lead Frontend Developer)**

My main contribution was leading the **Monorepo structure** and defining the component architecture for the Angular/Ionic clients. I concentrated on solving the **CORS integration challenges** and creating a uniform UX/UI foundation. This experience enhanced my skills in managing complex multi-platform frontends and prioritizing user experience in a security-first environment.

**Rafael Diaz (Data and Security Engineer)**

I was primarily responsible for enforcing the **ISO 27001 standard** by implementing the **JWT authentication middleware** and designing the **Entity-Relationship Model (E-R)**. The greatest challenge was ensuring that all data access was role-based and protected, making me proficient in secure API design and database migration using tools like **Alembic**.

**Group Reflection on Phase 2 Challenges**

The most significant technical challenge during the implementation phase was achieving **seamless and secure integration** between the three project components (Flask Backend, Angular Dashboard, Ionic App). Specifically, managing the **Cross-Origin Resource Sharing (CORS)** and configuring the **JWT token validation** across different platforms proved demanding. We overcame this by enforcing strict daily synchronization using Scrum, allowing us to debug integration issues in real-time. This phase confirmed the team's ability to translate high-level security requirements into concrete, robust software architecture, validating our initial choice of a decoupled, Monorepo approach.

# BIBLIOGRAFÍA

**Referencias Técnicas y de Plataforma**

Angular Team. (s.f.). *Angular Documentation*. Recuperado de<https://angular.dev/>.

Flask Development Team. (s.f.). *Welcome to Flask — Flask Documentation (3.1.x)*. Recuperado de<https://flask.palletsprojects.com/>.

Ionic Team. (s.f.). *Ionic Framework Documentation*. Recuperado de<https://ionicframework.com/developers>.

PostgreSQL Global Development Group. (s.f.). *Documentation - PostgreSQL*. Recuperado de<https://www.postgresql.org/docs/>.

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. Recuperado de<https://scrumguides.org/scrum-guide.html>.

**Estándares y Contexto**

IETF. (2015). *RFC 7519: JSON Web Token (JWT)*.

International Organization for Standardization (ISO). (2022). *ISO/IEC 27001:2022* (Referencia informativa sobre la versión vigente). Recuperado de [https://www.iso.org/obp/ui/#iso:std:iso-iec:27001:ed-4:v1:en](https://www.google.com/search?q=https://www.iso.org/obp/ui/%23iso:std:iso-iec:27001:ed-4:v1:en).

# ANEXOS

**Anexo A:** [**Diagrama Entidad-Relación (DER) del Modelo de Datos**](http://anexos/Anexo_A_Diagrama_ER.png)

* **Descripción:** Detalle de la estructura lógica de la base de datos PostgreSQL, incluyendo las entidades clave (USUARIO, PACIENTE, MÉDICO, HISTORIAL CLÍNICO) y sus relaciones, que justifican la integridad de datos y la escalabilidad del sistema.

**Anexo B:** [**Diagrama de Arquitectura Lógica de la Solución**](http://anexos/Anexo_B_Diagrama_de_Arquitectura.png)

* Descripción: Representación visual de la arquitectura desacoplada de MedInfo, ilustrando las tres capas principales (Frontend, Backend Flask y Almacenamiento Azure Database for PostgreSQL/Blob Storage) y el flujo de comunicación seguro entre ellas.

**Anexo C: Evidencias de Pruebas Funcionales (Capturas P1 a P4)**

* Descripción: Colección de capturas de pantalla de la terminal y la interfaz que validan los resultados de la implementación, incluyendo la conexión persistente a Azure Database for PostgreSQL, el registro de usuarios, la validación 401 del middleware JWT.